# Yale University
# Department of Computer Science


## Tight Bounds for the Sequence Transmission Problem


Da-Wei Wang  Lenore D. Zuck



YALEU/DCS/TR-705
May 1989

This work was supported in part by the National Science Foundation under gran
CCR-8405478.

# Tight Bounds for the Sequence Transmission Problem

Da-Wei Wang Lenore D. Zuck

**Abstract**

We investigate the problem of transmitting sequences over unreliable channels where both the data items and the message alphabet have finite domains. We show tight bounds on the number of different sequences that can be transmitted (as a function of size of the message alphabet) when the channel can (1) reorder and duplicate messages and (2) reorder and delete messages. All of our results are derived using formal reasoning about knowledge.

## 1 Introduction

One of the simplest and most basic problems of distributed computing is for one process, the *sender*, to reliably communicate a sequence of data items $x_0, x_1, \ldots$ to another process, the *receiver*. We call this the *sequence transmission problem (STP)*. Solving STP with a perfect channel—one that preserves message order and delivers each message within a known amount of time—is trivial: the sender simply sends each $x_i$ in turn. The receiver passively waits for each message and processes it when it arrives.

Real channels are not perfect. They may delay messages for arbitrary amounts of time, they may deliver messages out of order, and they may lose, duplicate, or corrupt messages. The *data link layer* ([BSW69,Car,Zim80]) in a standard protocol model attempts to solve STP under a particular set of assumptions about the underlying physical link layer (channel). Other common communication protocols such as virtual circuits, file transfer, and electronic mail are often built on top of this layer since the protocol designer

This work was supported in part by the National Science Foundation under gran CCR-8405478.

does not then have to be concerned about the faultiness of the physical channel.

Solutions to STP date back to the early work on communications protocols (cf. [BSW69,Ste76,AUY79,AUWY82]). Much of this early work was concerned with optimizing the number of states or the number of messages under various assumptions about the channel; for example, [AUY79, AUWY82] assume synchronous channels in which the loss of a message can be detected by the recipient at the next time step.

More recently, Lynch, Mansour, and Fekete [LMF88] considered asynchronous channels such as those we consider here. They proved that there is no "bounded-headers" data link layer protocol when the physical link can delete and reorder messages. Translated into our framework, this says roughly that there is no solution to STP using a finite message alphabet where reads and writes alternate (i.e., the sender reads a data element only after the receiver has written the previous one) and which satisfy a "boundedness" condition which we call here *weak boundedness*. Weak boundedness says that there is a number $k$ such that, no matter what has happened until the receiver has learnt the $i^{\text{th}}$ data element, it is possible that the $(i+1)^{\text{th}}$ data item is communicated to the receiver using at most $k$ messages.

In [AFWZ89] the problem is considered under the assumption that the message alphabet is finite and channels can reorder messages. It is shown there that if, in addition, the channel can duplicate messages then there is no solution to STP, even if the sender's protocol can depend non-uniformly on the data sequence (e.g., if the sender "knows" the entire input sequence in advance). For channels that can reorder and delete (but not duplicate) messages, a solution to STP is described which has the (undesirable) property that the number of steps it takes the receiver to "learn" the next data item depends on the past behavior of the channel.[1]

[AFWZ89] also consider restricted versions of STP, $\mathcal{X}$-STP, where there is a set $\mathcal{X}$ of possible input sequences which does not necessarily include all the sequences over the domain.

The above impossibility results imply that there are no practical solutions to $\mathcal{X}$-STP when $\mathcal{X}$ is uncountable. However, very often we wish to solve $\mathcal{X}$-STP only for finite sequences, thus, $\mathcal{X}$ is always countable. In this paper we address this question and consider $\mathcal{X}$-STP with *countable* $\mathcal{X}$ where message alphabets are finite and channels can reorder messages.

[1] The protocol does not satisfy the "boundedness" assumption of [LMF88] and therefore does not contradict their impossibility result.

We first show that if, in addition, the channel can duplicate messages, then there is no solution to $\mathcal{X}$-STP if there are more than $\alpha(m)$ sequences in $\mathcal{X}$, where $m$ denotes size of the sender's message alphabet and

$$\alpha(m) = m! \sum_{k=0}^{m} \frac{1}{k!}.$$

(As we show below, $\alpha(m)$ is the number of sequences over a domain of size $m$ that contain no repetitions.) This result holds even for the non-uniform case. We also show a solution to $\mathcal{X}$-STP(dup) (i.e., $\mathcal{X}$-STP with reorderings and duplications) in which $\mathcal{X}$'s size is exactly $\alpha(m)$.

We then turn to $\mathcal{X}$-STP(del), i.e., $\mathcal{X}$-STP with reorderings and deletions. We define *boundedness* and show that, just as in the case of $\mathcal{X}$-STP(dup), there is no bounded solution to $\mathcal{X}$-STP(del) if $|\mathcal{X}| > \alpha(m)$. A similar protocol to the one used in the reordering-duplication case solves $\mathcal{X}$-STP(del) for $|\mathcal{X}| = \alpha(m)$, so that this bound is also tight. Both impossibility results hold for the cases where the sender's protocol is allowed to depend non-uniformly on the input sequence.

We next discuss the definition of boundedness. For example, we show that for some countable $\mathcal{X}$ there is a weakly bounded solution to $\mathcal{X}$-STP(del). However, this solution is not bounded in any intuitive sense. In fact, it does not satisfy boundedness the way we define it.

We note that all the results reported here are derived using the knowledge viewpoint (for discussion and references, see [Hal87]). We consider this to be another demonstration of the power of reasoning about knowledge to analyze problems in distributed systems.

The paper is organized as follows: In Section 2 we present our formal framework and a formal statement of the problem. In Section 3 we discuss $\mathcal{X}$-STP(dup) and show a tight bound on $|\mathcal{X}|$ as a function of the size of the sender's alphabet. In section 4 we show similar results about bounded solution to $\mathcal{X}$-STP(del). In Section 5 we elaborate on boundedness and study some alternative definitions. We conclude in Section 6 with suggestions for future research.

# 2 Formal Model

## 2.1 Requirements of the Sequence Transmission Problem

In the *$\mathcal{X}$-sequence transmission problem* ($\mathcal{X}$-STP) there are two processors, the *sender* and the *receiver*, which communicate over a bidirectional commu-

nication link (the *channel*). There is an input tape with a (possibly infinite) sequence $X$, taken from a set $\mathcal{X}$ of *allowable* sequences over some finite domain $D$. The sender $S$ reads the data items of $X$ and tries to transmit them to the receiver $R$. $R$ must write these data items onto an output tape $Y$. We require that the following conditions hold for all $X \in \mathcal{X}$:

**Safety:** At any time, $Y$ is a prefix of $X$.

**Liveness:** If the channel satisfies appropriate *fairness* conditions, then every data element in the sequence $X$ is eventually written by $R$.

We consider here both channels that can reorder and delete messages, and channels that can reorder and duplicate messages. We term $\mathcal{X}$-STP with channels of first type *$\mathcal{X}$-STP(del)*, and $\mathcal{X}$-STP with channels of the second type—*$\mathcal{X}$-STP(dup)*.

Given $\mathcal{X}$, a *uniform* solution to $\mathcal{X}$-STP is a pair of protocols $(P_S, P_R)$ (for $S$ and $R$ respectively) such that for every $X \in \mathcal{X}$, $(P_S, P_R)$ started on input $X$ satisfies safety and liveness. A *non-uniform* solution to $\mathcal{X}$-STP is a family $\bigcup_{X \in \mathcal{X}}(P_{S,X}, P_R)$ of protocols such that for every $X \in \mathcal{X}$, $(P_{S,X}, P_R)$ satisfies safety and liveness.[2]

We restrict attention to non-probabilistic systems; since our correctness criteria are essentially deterministic (that is, we require every run to be correct), we lose no generality by this restriction.

The impossibility results reported here apply to non-uniform solutions to $\mathcal{X}$-STP where the number of distinct messages each of the processors can send is finite.

## 2.2 Modeling $\mathcal{X}$-STP Systems

We follow [HF88,HZ87] and model a distributed system as a set of *runs*, denoted by $\mathcal{R}$, each of which is an infinite sequence of *global states*, taken from some set $\mathcal{G}$ of global states. There is a set $\mathcal{G}_0 \subseteq \mathcal{G}$ of *initial global states*, and we assume that the first global state in every run is in $\mathcal{G}_0$. We restrict attention in this paper to distributed systems that solve $\mathcal{X}$-STP(dup) and $\mathcal{X}$-STP(del). A *global state* in such a system is a tuple of the form $(s_E, s_S, s_R)$, where $s_E$ denotes the *local state* of the environment, and $s_S$ (resp. $s_R$) denotes the local state of $S$ (resp. $R$). *Every* environment state encodes the

[2]Note that in this case the input tape is superfluous since we do not require our protocols to be finite state, and hence $P_{S,X}$ can have all of $X$ built into its code.

input sequence $X$ (to be read by $S$), the output sequence $Y$ (as written by $R$), and a list of which messages the environment can deliver.

A system is usually specified by a triple of protocols (algorithms), one for the sender, one for the receiver, and one for the environment (which in our case describes the possible behaviors of the channel). The system consists of all runs that are consistent with the protocols. The processor protocols are usually given explicitly, while the environment protocol is usually implicit.

We denote the (finite) set of message that that $S$ (resp. $R$) can send by $\mathcal{M}^S$ (resp. $\mathcal{M}^R$).

Let $r = r(0), r(1), \ldots$ be in $\mathcal{R}$. We denote the input sequence of $r$ by $X^r$. If $X^r$ is finite, i.e., $X^r = x_1, x_2, \ldots, x_k$, then we define its *length*, $|X^r|$, to be $k+1$. If $X^r$ is infinite, then we define $|X^r| = \omega$. If $X^r$ is empty then we define its length to be 1. For every $t \geq 0$, we refer to the pair $(r,t)$ as a *point*. We sometimes abuse notation and write $(r,t) \in \mathcal{R}$ to denote that $r \in \mathcal{R}$ and $t \geq 0$. We say that a run $r' = r'(0), r'(1), \ldots$ in $\mathcal{R}$, *extends* the point $(r,t)$, if for every $t' \leq t$, $r(t') = r'(t')$. Note that if $r'$ extends $(r,t)$ then $r$ extends $(r',t)$ and that $r'$ extends $(r,0)$; since we assume that the input sequence appears in every global state, it follows that $X^r = X^{r'}$.

When we say that a message is sent (resp. delivered, received) at a point $(r,t)$, we mean that the message is sent (resp. delivered, received) in the transition leading from $(r,t)$. For sake of simplicity, we assume that messages *cannot* be delivered in the same step they are sent. Thus, for example, since messages cannot be created by the channel, it follows that if a process receives a message at $(r,t)$ then this message must have been sent at $(r,t')$ for some $t' < t$. We also assume that at each point the channel can deliver to a process no more than one message.[3]

If $\mathcal{R}$ is a system that solves $\mathcal{X}$-STP(dup), then with every point $(r,t) \in \mathcal{R}$, we associate the vector, $dlvrble_R(r,t)$, (resp. $dlvrble_S(r,t)$) that stores, for every $\mu \in \mathcal{M}^S$ (resp. $\mu \in \mathcal{M}^R$), whether $\mu$ was sent before $(r,t)$, i.e.,

$$dlvrble_R(r,t)[\mu] = \begin{cases} 1 & \mu \text{ is sent to } R \text{ at } (r,t') \text{ for some } t' < t \\ 0 & \text{otherwise} \end{cases}$$

Similarly, if $\mathcal{R}$ solves $\mathcal{X}$-STP(del), then with every point $(r,t) \in \mathcal{R}$, we associate the vector, $dlvrble_R(r,t)$, (resp. $dlvrble_S(r,t)$) that stores, for every $\mu \in \mathcal{M}^S$ (resp. $\mu \in \mathcal{M}^R$), the number of copies of $\mu$ that were sent and not delivered by $(r,t)$, i.e.,

[3] All of our results hold without these assumptions.

$$dlvrble_R(r,t)[\mu] = |\{t' < t : \mu \text{ is sent to } R \text{ at } (r,t')\}| - |\{t' < t : \mu \text{ is delivered to } R \text{ at } (r,t')\}|.$$

Given two points $(r,t)$ and $(r',t')$ such that $r(t) = (s_E, s_S, s_R)$ and $r'(t') = (s'_E, s'_S, s'_R)$, and a process $p \in \{S,R\}$, we say that $p$ *cannot tell apart* $(r,t)$ *and* $(r',t')$, and denote it by $(r,t) \sim_p (r',t')$, if $s_p = s'_p$.

We want our system to be "sensible"; in particular, we assume that (a) $R$ does not know the input sequence at the beginning of a run, (b) the environment can arbitrarily delay messages and cannot discriminate between deliverable messages, and (c) if the system solves $\mathcal{X}$-STP(dup) then the environment cannot delete messages.

To capture (a) we assume that in all the initial states of the system $R$'s local state is the same. To capture (b) we require that for every point (i) there exists an extension where no message is delivered, and that (ii) for every deliverable message there is an extension where the message is delivered. To capture (c) we require that if $\mathcal{R}$ solves $\mathcal{X}$-STP(dup) then for every point $(r,t)$ in $\mathcal{R}$ there is a later point $(r,t')$ where every message that was sent before $(r,t)$ is delivered before $(r,t')$. Formally we require:

**Property 1**

**a.** *If both* $(s_E, s_S, s_R)$ *and* $(s'_E, s'_S, s'_R)$ *are in* $\mathcal{G}_0$, *then* $s_R = s'_R$.

**b.** *For every* $(r,t) \in \mathcal{R}$ *and* $p \in \{R,S\}$ *the following hold:*

- **i.** *There exists some* $r'$ *that extends* $(r,t)$ *such that no message is deliverd to* $p$ *at* $(r',t)$.
- **ii.** *For every* $\mu \in \mathcal{M}^p$ *such that* $\text{dlvrble}_{\bar{p}}(r,t)[\mu] > 0$, *there exists a run* $r_\mu$ *that extends* $(r,t)$ *such that* $\mu$ *is delivered to* $\bar{p}$ *at* $(r_\mu, t)$, *where* $\bar{p} = \{S,R\} - \{p\}$.

**c.** *If* $\mathcal{R}$ *solves* $\mathcal{X}$*-STP(dup), then for every point* $(r,t)$, *processor* $p \in \{R,S\}$, *and message* $\mu \in \mathcal{M}^p$ *there exists some* $t' > t$ *such that*

$$|\{t'' \le t' : \bar{p} \text{ receives } \mu \text{ at } (r,t'')\}| = |\{t'' \le t' : p \text{ sends } \mu \text{ at } (r,t'')\}|,$$

*where* $\bar{p} = \{S,R\} - \{p\}$.

## 2.3 Assigning Knowledge to Processors

We assume some set $\Phi$ of *basic facts* about the system, which includes $(\mathbf{x}_i = d)$ for every $i \geq 1$ and $d \in D$. We also assume an evaluation $\pi: \mathcal{G} \rightarrow 2^{\Phi}$ such that for every $g \in \mathcal{G}$, $\pi(g)$ includes all the basic facts that are true in $g$. We assume that $\pi$ is reasonable, so that, for example, if the $i^{\text{th}}$ element of $g$'s input sequence is 0, then $(\mathbf{x}_i = 0) \in \pi(g)$, and for no $d \neq 0$, is $(\mathbf{x}_i = d) \in \pi(g)$. We define a satisfiability relation $\models$ between sets of runs, points and basic facts such that $(\mathcal{R}, r, t) \models p$ iff $p \in \pi(r(t))$.[4]

The set of *facts* is the closure of the set of basic facts under the Boolean operators and the *knowledge* operator $K_p$ for $p \in \{S, R\}$. We extend the satisfiability relation to the set of facts in the natural way, viz. the semantics of $\models$ for a Boolean combination of facts is the usual, and for knowledge operators, we define

$$(\mathcal{R}, r, t) \models K_p\phi \quad \text{iff} \quad (\mathcal{R}, r', t') \models \phi \text{ for all } (r', t') \in \mathcal{R} \text{ such that } (r', t') \sim_p (r, t).$$

We use $K_p(\mathbf{x}_i)$ as an abbreviation for

$$\bigvee_{d \in D} K_p(\mathbf{x}_i = d),$$

read "$p$ knows the value of the $i^{\text{th}}$ input item".

The knowledge of a process depends, of course, on its local state; the more information stored in the local state of a process, the more it knows. If each processes stores (in its local state) its complete history, including the sequence of messages it has sent and received, then we say that the system has a *complete history interpretation* (see [HM84]). In *all* the impossibility proofs here we assume a complete history interpretation. Note that we are losing no generality in doing so.

Under the complete history interpretation, it is easy to see that $K_R(\mathbf{x}_i)$ is *stable*, that is, if $(\mathcal{R}, r, t) \models K_R(\mathbf{x}_i)$ then for every $t' \geq t$, $(\mathcal{R}, r, t') \models K_R(\mathbf{x}_i)$.

## 2.4 Formal Properties

Our safety requirements now translate to:

[4] The left hand side of the satisfiability relation should also include the evaluation function $\pi$; we, however, fix $\pi$ and omit explicit mention of it.

**Safety:** For every $r \in \mathcal{R}$ and $t \geq 0$, $(\mathcal{R}, r, t) \models (Y^r$ is a prefix of $X^r)$. (We assume that "$Y^r$ is a prefix of $X^r$" is a basic fact.)

The liveness requirement is more difficult to formulate as we require liveness only under "appropriate fairness conditions". We therefore assume that there is some set $\mathcal{F} \subseteq \mathcal{R}$ of *fair* runs. The only property we require from $\mathcal{F}$ is:

**Property 2** *For every point $(r,t) \in \mathcal{R}$ there exists a run $r' \in \mathcal{F}$ that extends $(r,t)$.*

This property guarantees that any prefix of a run can be extended to a fair run. It is satisfied by any notion of fairness that we are aware of, and it is the only property of fairness that we need for our impossibility results.

Let $\mathcal{F}$ be the set of fair runs in $\mathcal{R}$. We are now in a position to define $\mathcal{F}$-liveness, that is, liveness relative to $\mathcal{F}$:

**$\mathcal{F}$-Liveness:** For every $r \in \mathcal{F}$ and $i$, $1 \leq i < |X^r|$, there exists some $t \geq 0$ such that $(\mathcal{R}, r, t) \models (|Y^r| \geq i)$. (We assume that "$|Y^r| \geq i$" is a basic fact.)

Recall that, under the complete history interpretation, $K_R(\mathbf{x}_i)$ is stable for every $i$. We can therefore define, for every run $r \in \mathcal{F}$ and every $i < |X^r|$, the first time in $r$ where $R$ knows the values of the first $i$ data elements of $X^r$, and denote it by $t_i^r$. If no such time exists, we define $t_i^r$ to be $\infty$. That is, $t_i^r$ is the minimal $t$ such that:

$$(\mathcal{R}, r, t) \models \bigwedge_{j=1}^{i} K_R(\mathbf{x}_j)$$

if for some $t \geq 0$, $(\mathcal{R}, r, t) \models \bigwedge_{j=1}^{i} K_R(\mathbf{x}_j)$, or $\infty$ otherwise. Note that for every $r \in \mathcal{F}$, $t_i^r < \infty$ for every $i$, $1 \leq i < |X^r|$.

Although we mentioned before that our results are derived using knowledge arguments, we shall hardly mention knowledge from now on. However, as we shall see, the $t_i$'s we've just defined play a major role in our proofs, and these $t_i$'s are defined in terms of knowledge. The $t_i$'s capture when $R$ "learns" the $i^{\text{th}}$ data item, assuming it has learnt the previous $i-1$ data items. The obvious intuitive definition seems to be "$R$ learns the $i^{\text{th}}$ data item when it receives a message containing that data item". This, however, is not well defined since it is possible that no particular message contains

this information. An alternative definition might be "$R$ learns the $i^{\text{th}}$ data item just before it writes it". But here again, it is possible to design protocols where $R$ writes the $i^{\text{th}}$ data item well after $R$ has learnt it. For example, $S$ can send $R$ a single message which informs $R$ the values of several data items, and there is no way $R$ can write them at the same step. We therefore feel that the right definition of the $t_i$'s is in terms of knowlege.

# 3 $\mathcal{X}$-STP(dup)

In this section we show that $\mathcal{X}$-STP(dup) is unsolvable if $|\mathcal{X}| > \alpha(|\mathcal{M}^S|)$. Intuitively, in $\mathcal{X}$-STP(dup), at every step the channel can deliver a copy of any message that had been sent in the past.

Fix a set $\mathcal{X}$ of sequences and a system $\mathcal{R}$ that solves $\mathcal{X}$-STP(dup). Let $m$ be $|\mathcal{M}^S|$. It seems quite easy to see that $|\mathcal{X}| \leq \alpha(m)$: Once $S$ sends some message $\mu$, the channel can deliver to $R$ an unbounded number of copies of $\mu$. Hence, $S$ could gain nothing by sending more than one copy of each message. The number of sequences over $\mathcal{M}^S$ that contain no repetitions is

$$\sum_{k=0}^{m} \binom{m}{k} k! \;=\; \sum_{k=0}^{m} \frac{m!k!}{k!(m-k)!} \;=\; m! \sum_{k=0}^{m} \frac{1}{k!} \;=\; \alpha(m)$$

and the result follows.

It is not clear how to formalize the intuition above, since $R$ may be able to tell apart points in which it has received the same message sequences, for example, by the time (on $R$'s local clock) that the messages are received or by the sequence of messages $R$ itself has sent. Hence, a straightforward formalization of the intituition is bound to fail. We therefore take another approach, namely, we show that if $|\mathcal{X}| > \alpha(m)$ then some run violates the safety property. To this end, we assume that $|\mathcal{X}| > \alpha(m)$ and take $\alpha(m)+1$ runs whose input sequences are mutually distinct. We then proceed, by induction, to show that for all $\ell = 0, \ldots, m$ there are "enough" $(\alpha(m-\ell)+1)$ points whose input sequences are mutually distinct and that $R$ cannot tell apart, such that before each of them $S$ has sent every message from some set of $\ell$ messages. We call such a set points and set of messages a *dup-decisive tuple*. When $\ell = m$, we obtain two points with different input sequences that $R$ cannot tell apart and by which $S$ has sent every message in $\mathcal{M}^S$. It is now easy to show that at least one of these points has an extension which violates the safety propery.

We start with a formal definitions of dup-decisive tuples

**Definition 1** *A* dup-decisive tuple $\Gamma$ *is a tuple* $\langle \mathcal{R}', t, M \rangle$ *where* $\mathcal{R}' \subseteq \mathcal{F}$, $t \geq 0$, $M \subseteq \mathcal{M}^S$ *and for every run* $r \in \mathcal{R}'$, *the following all hold:*

1. *For every* $\mu \in M$, $\text{dlvrble}_R(r,t)[\mu] = 1$.
2. *For every* $r' \in \mathcal{R}'$, $(r,t) \sim_R (r',t)$ *and if* $r' \neq r$ *then* $X^r \neq X^{r'}$.

*That is, a dup-decisive tuple consists of a set of points, belonging to fair runs, that R cannot tell apart whose input sequences are mutually distinct, such that before each of these points S has sent all the messages in M.*

The following lemma is the main lemma in this section. It says that if there is a dup-decisive tuple $\Gamma = \langle \mathcal{R}', t, M \rangle$ with $|\mathcal{R}'| \geq 2$, then for every run $r \in \mathcal{R}'$ whose input sequence is not a prefix of all the others, some message not in $M$ is delivered to $R$ at or after the $t^{\text{th}}$ step.

**Lemma 1** *Let* $\Gamma = \langle \mathcal{R}', t, M \rangle$ *be a dup-decisive tuple with* $|\mathcal{R}'| \geq 2$. *Let* $r \in \mathcal{R}'$ *be such that for some* $r' \in \mathcal{R}'$, $X^r$ *is not a prefix of* $X^{r'}$. *Then there exists a message* $\mu \notin M$ *that is delivered to R at* $(r,t')$ *for some* $t' \geq t$.

**Proof:** Assume, by way of contradiction, that at every point $(r,t')$, $t' \geq t$, every message that is delivered to $R$ is in $M$. It follows from Property 1 that for every run $r' \in \mathcal{R}'$, there exists a run $r'_e$ that extends $(r',t)$ such that for all points $(r'_e,t')$, $t' \geq 0$, $(r'_e,t') \sim_R (r,t')$. Since $r$ is fair, it follows that for every $i < |X^r|$, $t^r_i < \infty$. Thus, for every $r' \in \mathcal{R}'$, $i < |X^r|$, and $d \in D$, $(\mathcal{R}, r, t^r_i) \models K_R(\mathbf{x}_i = d)$ implies that $(\mathcal{R}, r'_e, t^r_i) \models K_R(\mathbf{x}_i = d)$, and therefore that $(\mathcal{R}, r', t) \models (\mathbf{x}_i = d)$. Consequently, $X^r$ is a prefix of $X^{r'}$ for all $r' \in \mathcal{R}'$, contradicting our assumption. ■

Lemma 1 implies that if $\langle \mathcal{R}', t, M \rangle$ is a dup-decisive tuple such that $|\mathcal{R}'| \geq 2$, then we can take a run $r \in \mathcal{R}'$ (whose input sequence is not a prefix of all the others), and find the minimal $t' \geq t$ such that some message $\mu \notin M$ is delivered to $R$ at $(r,t')$. Since every message that is delivered to $R$ before $(r,t)$ is in $M$ and was therefore sent in each of $\mathcal{R}'$'s runs before the $t^{\text{th}}$ point, we can find, for each run $r' \in \mathcal{R}'$, an extensions $r''$ of the $(r,t)$ points such that $R$ cannot tell apart $(r',t')$ and $(r'',t')$. This implies that there exists a dup-decisive tuple $\langle \mathcal{R}'', t', M \rangle$ such that the runs in $\mathcal{R}''$ are all extensions of the $t^{\text{th}}$ points of $\mathcal{R}'$'s runs and for at least one run (namely $r$) in $\mathcal{R}''$, some message not in $M$ is sent before the $(t')^{\text{th}}$ point.

We can repeat the same construction to derive a dup-decisive tuple $\langle \mathcal{R}''', t'', M \rangle$ such that the runs in $\mathcal{R}'''$ are all extensions of the $t^{\text{th}}$ point

of $\mathcal{R}''$s runs and for at least two run in $\mathcal{R}''$, some message not in $M$ is sent before the $(t'')^{\text{th}}$ step.

The construction can be repeated ($|\mathcal{R}'|-1$ times) until some dup-decisive tuple $\langle \mathcal{R}^*, t^*, M \rangle$ is obtained where in all but possibly one run $r \in \mathcal{R}^*$, at some step before $t^*$, some message not in $M$ is sent to $R$.

This leads us to:

**Corollary 1** *Let $\Gamma = \langle \mathcal{R}', t, M \rangle$ be a dup-decisive tuple with $|\mathcal{R}'| \geq 2$. Then there exists a dup-decisive tuple $\Gamma' = \langle \mathcal{R}'', t', M \rangle$ such that the following all hold:*

1. *$|\mathcal{R}''| = |\mathcal{R}'|$ and $t' \geq t$.*
2. *For every $r \in \mathcal{R}'$ there exists a run $r' \in \mathcal{R}''$ such that $r'$ extends $(r, t)$.*
3. *$|\{r \in \mathcal{R}'' : \text{dlvrble}_R(r, t')[\mu] = 1$ for some $\mu \notin M\}| = |\mathcal{R}'| - 1$.*

**Lemma 2** *If $|\mathcal{X}| > \alpha(m)$ then for all $\ell \leq m$ there exists a dup-decisive tuple $\Gamma_\ell = \langle \mathcal{R}_\ell, t_\ell, M_\ell \rangle$ such that $|\mathcal{R}_\ell| = \alpha(m-\ell)+1$ and $|M_\ell| = \ell$.*

**Proof:** The proof is by induction on $\ell$. For the base case we let $\mathcal{R}_0$ be any set of $\alpha(m)+1$ runs whose input sequences are mutually distinct, let $t_0$ be 0, and let $M_0$ the empty set. For the inductive step, assume that $\Gamma_\ell = \langle \mathcal{R}_\ell, t_\ell, M_\ell \rangle$ is defined for some $\ell < m$. From Corollary 1 it now follows that there exists a dup-decisive tuple

$$\Gamma^*_\ell = \langle \mathcal{R}^*_\ell, t^*_\ell, M_\ell \rangle$$

such that the following all hold:

1. $|\mathcal{R}^*_\ell| = |\mathcal{R}_\ell|$ and $t^*_\ell \geq t_\ell$.
2. For every $r \in \mathcal{R}_\ell$ there exists a run $r' \in \mathcal{R}^*_\ell$ such that $r'$ extends $(r, t_\ell)$.
3. $|\{r \in \mathcal{R}^*_\ell : dlvrble_R(r, t^*_\ell)[\mu] = 1$ for some $\mu \notin M_\ell\}| \geq \alpha(m-\ell)$.

Let $\mathcal{R}'_\ell$ denote the set of runs in $\mathcal{R}^*_\ell$ that satisfy (3). For every run $r$ in $\mathcal{R}'_\ell$, there exists some message $\mu_r \notin M_\ell$ such that $dlvrble_R(r, t^*_\ell)[\mu_r] = 1$. It thus follows that for some $\mu \in \mathcal{M}^S - M_\ell$, for at least

$$\frac{|\mathcal{R}'_\ell|}{m-\ell} \geq \frac{\alpha(m-\ell)}{m-\ell} > \alpha(m-(\ell+1))$$

runs $r \in \mathcal{R}'_\ell$, $\mu_r = \mu$. Let $\mathcal{R}_{\ell+1}$ be a set of $\alpha(m-(\ell+1))+1$ of those runs. We can now define

$$\Gamma_{\ell+1} = \langle \mathcal{R}_{\ell+1}, t^*_\ell, M_\ell \cup \{\mu\} \rangle,$$

which completes the inductive step. ■

**Corollary 2**

$$|\mathcal{X}| \leq \alpha(m).$$

**Proof:** Assume to the contrary that $|\mathcal{X}| > \alpha(m)$. From Lemma 2 it follows that there exists a dup-decisive tuple $\langle \mathcal{R}', t, \mathcal{M}^S \rangle$ where $|\mathcal{R}'| = \alpha(0)+1 = 2$. It thus follows from Lemma 1 that there exists a run $r \in \mathcal{R}'$ such that some message $\mu \notin \mathcal{M}^S$ is delivered to $R$ at $(r, t')$ for some $t' \geq t$, which is obviously a contradiction. ■

We therefore have:

**Theorem 1** *Let $\mathcal{X}$ be a set of seuquences and let $\mathcal{R}$ be a system that solves $\mathcal{X}$-STP(dup) where $|\mathcal{M}^S| = m$. Then*

$$|\mathcal{X}| \leq \alpha(m).$$

We are naturally led to ask whether the bound of Theorem 1 can be improved. We conclude this section by showing how the intuitive arguments at the beginning of this section lead to a solution of $\mathcal{X}$-STP(dup) where $|\mathcal{X}| = \alpha(m)$, and thus establish that the above bound is tight.

Assume $D = \{d_1, \ldots, d_m\}$ and let $\mathcal{X}$ be the set of sequences over $D$ that have no repetitions of data items. Consider now the following protocol where $\mathcal{M}^S = \{d_1, \ldots, d_m\} = \mathcal{M}^R$. $S$ sends the data items in sequence and waits for the appropriate acknowledgements for each. $R$ awaits the arrival of some *new* message (i.e., one different than any of the previously received messages); it then writes the new data item and sends the appropriate acknowledgement to $S$. Hence, reordering is dealt with by simply allowing the processors to ignore previously received messages. Note that the protocol is finite state.

It is easy to see that the system generated by the protocol satisfies the safety property. To see that it satisfies the liveness property, we define a fair run to be any run in which every message that is sent is eventually delivered. Since we require (in Property 1, part c) that in every run every message that is sent is eventually delivered, it follows that every run is fair. It is now easy to see that the system satisfies the liveness propery as well.

This gives an example of one particular set $\mathcal{X}$ of sequences for which there is a protocol such that $|\mathcal{X}| = \alpha(m)$ which solves $\mathcal{X}$-STP(dup). One can show that, given a set $\mathcal{X}$, in order to solve $\mathcal{X}$-STP(dup) it is necessary to map every input sequence $X \in \mathcal{X}$ to a message sequence $\mu(X)$ over $\mathcal{M}^S$ such that $\mu(X)$ has no repetitions and for every $X_1, X_2$ in $\mathcal{X}$, $\mu(X_1)$ is a prefix of $\mu(X_2)$ only when $X_1$ is a prefix of $X_2$. When $|\mathcal{X}| \leq m!$ one can always find such a mapping; if the sequences in $\mathcal{X}$ are such that some are prefix of the others, then one can do better, but no better than $|\mathcal{X}| = \alpha(m)$.

# 4 $\mathcal{X}$-STP(del)

In this section we study $\mathcal{X}$-STP(del). Intuitively, in $\mathcal{X}$-STP(del), at every step the channel can deliver a copy of any message that was sent and was not delivered in the past. In order to model this, the environment stores, in its local state, how many copies of each message were sent and not yet delivered. It then arbitrarily delivers available copies of messages (and updates their count).

In [AFWZ89] an "unbounded" solution to $\mathcal{X}$-STP(del) is given; roughly speaking, a solution is unbounded if the number of steps it takes $R$ to learn a new data item depends on the history of the run. A solution to $\mathcal{X}$-STP is *$f$-bounded*, where $f$ is some function, if for every run $r$ and $i < |X^r|$, if $t > t^r_{i-1}$ then then there exists a run $r_t$ that extends $(r,t)$ such that

$$t^{r_t}_i \leq t + f(i).$$

Moreover, we want $r_t$ not to depend on any long lost message, so we also require that the channel does not deliver any message that was sent prior to $(r,t)$ at any $(r_t, t')$, $t \leq t' < t^{r_t}_i$. Formally:

**Definition 2** *Let $f$:$\mathbf{N} \to \mathbf{N}$ be some function. A system $\mathcal{R}$ that solves $\mathcal{X}$-STP(del) is* $f$-bounded, *or simply* bounded, *if for every run $r \in \mathcal{R}$, $t \geq 0$, and $i < |X^r|$, if $i = 1$, or if $i > 0$ and $t > t^r_{i-1}$, then there exists a run $r_t$ that extends $(r,t)$ such that the following all hold:*

1. $t^{r_t}_i \leq t + f(i)$.
2. *The only messages delivered in $r_t$ between $t$ and $t^{r_t}_i$ are messages that were sent $i$ that interval, i.e., for every $t'$, $t < t' \leq t^{r_t}_i$ and $p \in \{S, R\}$,* $\text{dlvrble}_p(r_t, t') \geq \text{dlvrble}_p(r_t, t)$ *(where $\geq$ is defined in the obvious way).*

We say that $\mathcal{R}$ is *unbounded* if it is not $f$-bounded for every function $f$.

Our goal is to show that there are no bounded solutions to $\mathcal{X}$-STP(del) if $|\mathcal{X}| > \alpha(m)$, where $m = |\mathcal{M}^S|$ is defined as before. While in the previous section the intuitive justification of the similar result is straightforward, here, we think, the result is rather suprising.

Fix a set $\mathcal{X}$ of sequences, a function $f: \mathbf{N} \to \mathbf{N}$, and $f$-bounded system $\mathcal{R}$ that solves $\mathcal{X}$-STP(del). Let $m = |\mathcal{M}^S|$. The proof that $|\mathcal{X}| > \alpha(m)$ is similar to the corresponding proof in Section 4. There are, however, several differences, some due to the difference in the channel behavior and some due to the boundedness assumption. To prove Theorem 1 we defined dup-decisive tuples which consist of a set $\mathcal{R}'$ of fair runs whose input sequences are mutually disjoint, some $t \geq 0$ such that $R$ cannot tell apart the $t^{\text{th}}$ points of the runs in $\mathcal{R}'$, and a set of messages $M \subseteq \mathcal{M}^S$ such that each message in $M$ is sent before every $(r,t)$, $r \in \mathcal{R}'$. We then claimed, in Lemma 1, that in all but possibly one run in $\mathcal{R}'$, some message not in $M$ is delivered to $R$. The proof relied heavily on the fact that in the duplication case a message that was sent can be delivered arbitrarily many times. In the deletion case a message can only be delivered if it was sent more times than it was delivered, so the similar claim cannot hold in the deletion case. However, if we add to the definition of dup-decisive tuples some counter $n$ and require that each message in $M$ is sent at least $n$ times more than it is delivered before every $(r,t)$, $r \in \mathcal{R}'$, then if we choose $n$ and the runs in $\mathcal{R}'$ carefully, a similar claim should hold. We therefore define del-decisive tuples as follows:

**Definition 3** *A* del-decisive tuple $\Gamma$ *is a tuple* $\langle \mathcal{R}', t, M, n \rangle$, *where* $\mathcal{R}' \subseteq \mathcal{F}$,$t \geq 0$, $M \subseteq \mathcal{M}^S$, $n \geq 0$, *and for every run* $r \in \mathcal{R}'$, *the following all hold:*

1. *For all* $\mu \in M$, $\text{dlvrble}_R(r,t)[\mu] \geq n$.
2. *For every* $r' \in \mathcal{R}'$, $(r,t) \sim_R (r',t)$ *and if* $r' \neq r$ *then* $X^r \neq X^{r'}$.

In order to derive a lemma similar to Lemma 1, we have to choose some $c \geq 0$ such that we are guaranteed that for all but possibly one run $r \in \mathcal{R}'$, in some fair extension of $r$, some message not in $M$ is delivered to $R$ within at most $c$ steps after $t$. We derive $c$ from the the boundedness assumption: If $\beta$ is such that all the (finitely many) runs in $\mathcal{R}'$ can be uniquely identified by their $\beta$-prefix, $c$ can be defined as $\sum_{i=1}^{\beta} f(i)$. That is, if we consider only "efficient" extensions of the $t^{\text{th}}$ points in $\mathcal{R}'$'s runs, namely, extensions in which $R$ learns the $i^{\text{th}}$ data item within $f(i)$ steps, then the boundedness assumption guarantees that for all but possibly one of the run in $\mathcal{R}'$, within $c$ steps some message not in $M$ is delivered to $R$.

an extension of $r''$ of $r'$ such that $R$ cannot tell apart $(r',t')$ and $(r'',t')$. This implies that there exists a del-decisive tuple $\langle \mathcal{R}'', t', M, n-c\rangle$ such that the runs in $\mathcal{R}''$ are all extensions of the $t^{\text{th}}$ points of $\mathcal{R}''$s and for at least one run (namely $r'$) in $\mathcal{R}''$, some message $\mu \notin M$ is sent more times than it is delivered by the $(t')^{\text{th}}$ step, i.e., for some $r \in \mathcal{R}''$ and $\mu \notin M$, $dlvrble_R(r,t'')[\mu] \geq 1$.

If $n \geq 2c$ we can repeat the same construction to derive a del-decisive tuple $\langle \mathcal{R}''', t'', M, n-2c\rangle$ such that the runs in $\mathcal{R}'''$ are all extensions of the $t^{\text{th}}$ point of $\mathcal{R}''$s runs, and for one $r \in \mathcal{R}'''$, $\sum_{\mu \notin M} dlvrble_R(r,t'')[\mu] \geq 2$. Alternatively, we can derive a del-decisive tuple $\langle \mathcal{R}''', t'', M, n-2c\rangle$ such that the runs in $\mathcal{R}'''$ are all extensions of the $t^{\text{th}}$ point of $\mathcal{R}''$s runs, and for at least two run in $r \in \mathcal{R}''$, for some $\mu \notin M$, $dlvrble_R(r,t'')[\mu] \geq 1$.

If $n \geq n'(|\mathcal{R}'|-1)$ for some $n' \geq 0$, then the construction can be repeated, $n'(|\mathcal{R}'|-1)$ times, until some del-decisive tuple $\langle \mathcal{R}^*, t^*, M, n-n'(|\mathcal{R}'|-1)c\rangle$ is obtained where in all but possibly one run $r \in \mathcal{R}^*$, $\sum_{\mu \notin M} dlvrble_R(r,t'')[\mu] \geq n'$.

This leads us to the deletion equivalent of Corollary 1:

**Corollary 3** *Let $\langle \mathcal{R}', t, M, n\rangle$ is del-decisive tuplewhere $|\mathcal{R}'| \geq 2$ and $\bigcup_{r\in\mathcal{R}'} \subseteq \mathcal{X}'$. Then, for every $n' \geq 0$, if $n \geq n'c(|\mathcal{R}'|-1)$, then there exists a del-decisive tuple $\langle \mathcal{R}'', t', M, n-n'c(|\mathcal{R}'|-1)\rangle$ such that the following all hold:*

1. *$|\mathcal{R}''| = |\mathcal{R}'|$ and $t' \geq t$.*
2. *For every $r \in \mathcal{R}'$ there exists a run $r' \in \mathcal{R}''$ such that $r'$ extends $(r,t)$.*
3. *For $|\mathcal{R}'|-1$ runs $r \in \mathcal{R}''$,*

$$\sum_{\mu \notin M} \text{dlvrble}_R(r,t')[\mu] \geq n',$$

*that is, by $(r,t')$ there are at least $n'$ copies of messages not in $M$ that are sent and not delivered to $R$.*

Our goal is obviously to show that if $|X| > \alpha(m)$, then the channel can obtain at least $c$ copies of every message in $\mathcal{M}^S$, and thus derive that there must be runs that violate the safety property. To this end, we define a sequence $\{\delta_i\}_{i=0}^m$, such that $\delta_i$ is the number of copies that suffices for the channel to obtain of each of $i$ messages in order to be able to obtain $\delta_{i+1}$ copies of each of $i+1$ messages. The $\delta_i$ are defined recursively, where

$$\delta_m = c$$

To define $\beta$, we first fix some subset $\mathcal{X}'$ of $\mathcal{X}$ whose size is $\min(|\mathcal{X}|, \alpha(m)+1)$ and define $\beta$ be the minimal $i$ such that every sequences in $\mathcal{X}'$ can be uniquely identified by its $i^{\text{th}}$ prefix.

We next define the "efficient extensions" above, termed here $\beta$-extensions. A *$\beta$-extension of a point* $(r,t)$ is an extension of $(r,t)$ in which $R$ is guaranteed to learn all the first $\beta_r$ data items within at most $c = \sum_{i=1}^{\beta_r} f(i)$ steps after $t$ without having received any message that was sent prior to $(r,t)$, where $\beta_r = \min(\beta, |X^r| - 1)$. Formally:

**Definition 4** *Let $(r,t) \in \mathcal{R}$ and let $\beta_r = \min(\beta, |X^r| - 1)$. We say that a run $r' \in \mathcal{R}$ is a* $\beta$-extension *of $(r,t)$ if the following all hold:*

1. *$r'$ extends $(r,t)$.*
2. *$t^{r'}_{\beta_r} \leq t + \sum_{i=1}^{\beta} f(i)$.*
3. *For all $t'$, $t \leq t' \leq t^{r'}_{\beta_r}$, $\text{dlvrble}_R(r',t') \geq \text{dlvrble}_R(r,t')$.*

Note that every point $(r,t) \in \mathcal{R}$ has some $\beta$-extension.

The following lemma is similar to Lemma 1. It shows that if there is a del-decisive tuple $\langle \mathcal{R}', t, M, n \rangle$ such that $n \geq c$ and all the input sequences to the runs $\mathcal{R}'$ are in $\mathcal{X}'$, then, for every run $r \in \mathcal{R}'$ whose input sequence is not a prefix to all the others, in every $\beta$-extension $r'$ of $(r,t)$, some message not in $M$ is delivered within $c$ steps after $t$.

**Lemma 3** *Let $\Gamma = \langle \mathcal{R}', t, M, n \rangle$ be a del-decisive tuple with $|\mathcal{R}'| \geq 2$, $n \geq c$, and $\bigcup_{r \in \mathcal{R}'} X^r \subseteq \mathcal{X}'$. Let $r \in \mathcal{R}'$ be such that for some $r' \in \mathcal{R}'$, $X^r$ is not a prefix of $X^{r'}$. Then, in every $\beta$-extension $r'$ of $(r,t)$, there exists some $\mu \notin M$ such that $\mu$ is delivered to $R$ at some $(r',t')$, $t \leq t' < t + c$.*

**Proof:** The proof is similar to the proof of Lemma 1 and left to the reader. ■

Lemma 3 implies that if $\langle \mathcal{R}', t, M, n \geq c \rangle$ is a del-decisive tuple such that $|\mathcal{R}'| \geq 2$ and the input sequences of all of $\mathcal{R}'$'s runs are in $\mathcal{X}'$, then for every run $r \in \mathcal{R}'$ whose input sequence is not a prefix of all the others, in all of $r$'s $\beta$-extensions some message not in $M$ is delivered to $R$ before the $(t+c)^{\text{th}}$ step. We can therefore take some $\beta$-extension $r'$ of $(r,t)$ and find the first $(r',t')$ in $r'$, $t' \geq t$, at which some message $\mu \notin M$ is delivered to $R$.

Since $t' < t + c$, and since every message delivered between $(r',t)$ and $(r',t')$ is in $M$ and was therefore sent at least $n$ times more that it was delivered in by each $(r',t)$, $r' \in \mathcal{R}'$, we can find, for each run $r' \in \mathcal{R}'$,

and for every $\ell < m$,

$$\delta_\ell = \delta_{\ell+1}(1 + c(m-\ell)\alpha(m-\ell)).$$

We are now ready to prove the equivalent of Lemma 2.

**Lemma 4** *If $|\mathcal{X}'| > \alpha(m)$ then for all $\ell = 0, \ldots, m$, there exists a del-decisive tuple $\Gamma_\ell = \langle \mathcal{R}_\ell, t_\ell, M_\ell, \delta_\ell \rangle$ such that $|\mathcal{R}_\ell| = \alpha(m-\ell)+1$ and $|M_\ell| = \ell$.*

**Proof:** The proof is by induction on $\ell$. The base case is trivial. For the inductive step, assume that $\Gamma_\ell = \langle \mathcal{R}_\ell, t_\ell, M_\ell, \delta_\ell \rangle$ ($\ell < m$) is defined. While in the duplication case all we have to do at this point is to show that every run in $\mathcal{R}_\ell$ can be extended so that some new message (not in $M_\ell$) is sent, here we want to find extensions in which $\delta_{\ell+1}$ copies of some new message are sent.

Let $n_\ell = \delta_{\ell+1}(m-\ell)$, so that

$$\delta_\ell - \delta_{\ell+1} \;=\; \delta_{\ell+1}(m-\ell)c\alpha(m-\ell) \;=\; n_\ell c(|\mathcal{R}_\ell| - 1).$$

It follows now from Corollary 3 that there exists a del-decisive tuple

$$\Gamma^*_\ell = \langle \mathcal{R}^*_\ell, t^*_\ell, M_\ell, \delta_{\ell+1} \rangle$$

such that the following all hold:

1. $|\mathcal{R}^*_\ell| = |\mathcal{R}_\ell|$ and $t^*_\ell \geq t_\ell$.
2. For every $r \in \mathcal{R}_\ell$ there exists a run $r' \in \mathcal{R}^*_\ell$ such that $r'$ extends $(r, t_\ell)$.
3. For $\alpha(m-\ell)$ runs $r \in \mathcal{R}^*_\ell$,

$$\sum_{\mu \notin M_\ell} dlvrble_R(r, t^*_\ell)[\mu] \geq n_\ell.$$

Let $\mathcal{R}'_\ell$ denote the set of runs in $\mathcal{R}^*_\ell$ that satisfy (3). For every run $r$ in $\mathcal{R}'_\ell$, by $(r, t^*)$ there are at least $n_\ell$ copies of message not in $M_\ell$ that were sent and not delivered. Since $|\mathcal{M}^S - M_\ell| = (m-\ell)$, if follows that for every run $r \in \mathcal{R}'_\ell$, there exists some $\mu_r \notin \mathcal{M}$ such that

$$dlvrble_R(r, t^*_\ell)[\mu_r] \geq \frac{n_\ell}{m-\ell} = \delta_{\ell+1}.$$

It thus follows that for some $\mu \in \mathcal{M}^S - M_\ell$, for at least

$$\frac{|\mathcal{R}'_\ell|}{m-\ell} \geq \frac{\alpha(m-\ell)}{m-\ell} > \alpha(m-(\ell+1))$$

runs $r \in \mathcal{R}'_\ell$, $\mu_r = \mu$. Let $\mathcal{R}_{\ell+1}$ be a set of $\alpha(m-(\ell+1))+1$ of these runs. We define

$$\Gamma_{\ell+1} = \langle \mathcal{R}_{\ell+1}, t^*_\ell, M_\ell \cup \{\mu\}, \delta_{\ell+1} \rangle.$$

■

We therefore have:

**Corollary 4** $|\mathcal{X}| \leq \alpha(m)$.

**Proof:** Assume to the contrary that $|\mathcal{X}| > \alpha(m)$. From Lemma 4 it follows that there exists a del-decisive tuple $\langle \mathcal{R}', t, \mathcal{M}^S, c \rangle$ where $|\mathcal{R}'| = 2$. It thus follows from Lemma 3 that there exists a run $r \in \mathcal{R}'$ such that for some $\beta$-extention $r'$ of $(r,t)$, some message $\mu \notin \mathcal{M}^S$ is delivered at $(r',t')$ for some $t' \geq t$, which is a contradiction. ■

Corollary 4 implies:

**Theorem 2** *Let $\mathcal{X}$ be a set of seuquences and let $\mathcal{R}$ be a bounded system that solves $\mathcal{X}$-STP(del) where $|\mathcal{M}^S| = m$. Then*

$$|\mathcal{X}| \leq \alpha(m).$$

We conclude this section by pointing out that the solution to $\mathcal{X}$-STP(dup) with $|\mathcal{X}| = \alpha(m)$ described at the end of Section 3 can easily be modified to give a bounded solution to $\mathcal{X}$-STP(del) with $|\mathcal{X}| = \alpha(m)$, so that $\alpha(m)$ is a tight bound on the size of $\mathcal{X}$.

# 5 More About Boundedness

In [LMF88] it is shown that there are no "$k$-bounded Data Link Layer Protocols that use finitely many headers". Translated into our formalism, this roughly means that there are no uniform '$k$-bounded' solutions to $D^\omega$-STP(del) if the processors' message alphabets are finite, where $k$ is some constant function. The boundedness requirement in [LMF88] is however different than the boundedness presented here; we call it here *weak boundedness*. Intuitively, while in a bounded system every point has a bounded

extension, in a weakly bounded system only the $(t_i^r)^{\text{th}}$ points of runs have bounded extensions.

The results of [AFWZ89] show that $\mathcal{X}$-STP(del) is solvable, though the solution in not bounded in the sense of [LMF88]. It is also conjectured there that $\mathcal{X}$-STP has no weakly-bounded solutions if $\mathcal{X}$ is uncountable.

Boundedness is an attempt to measure a protocol in terms of how fast it recovers from faults, so that an unbounded protocol is one that never fully recovers. Our motivation in strengthening the [LMF88,AFWZ89] definition of boundedness is that weak boundedness allows for some impractical solutions to $\mathcal{X}$-STP(del). For example, it allows protocols that include runs $r$ where a single fault occurs, say right after $t_i^r$, and yet there is no bound on $t_{i+1}^{r'}$ for any extesion $r'$ of $(r, t_i^r + 1)$ which depends only on $i$.

For example, let $\mathcal{X}$ be the set of all finite sequences over $D$ (so that $\mathcal{X}$ is a countable set) and consider the following protocol: $S$ transmits the data items in sequence and $R$ writes and acknowledges them using a Alternating Bit protocol (ABP), until one of the processors fails to receive a message in time. (We are assuming here some global clock and known message delivery times.) This processor then starts to execute the [AFWZ89] protocol, using a different message alphabet than that used in the initial part of the execution. In the [AFWZ89] protocol, $S$ reads the whole input sequence and transmits the data items in *reverse* order. Thus, after having learnt some prefix of the sequence, $R$ starts to learn some of its suffix. If the old lost message is delivered, the processors resume execution of the original protocol. Thus, the processors alternate between executions of the ABP where $R$ learns the prefix of the sequence, and executions of the [AFWZ89] protocol where $R$ learns the suffix of the sequence, until $S$ sends a special message indicating to $R$ that the prefix and the suffix learnt consist of the whole sequence.

To see why this protocol is $k$-bounded, note that new $t_i$'s can be obtained only during an execution of the ABP or by the delivery of the special message indicating the end of the execution. In the former case, $k$-boundedness is obvious; in the latter, when $t_i$ is obtained, so are all the $t_j$'s for every $j \geq i$. Hence, if only one fault occurs in a run, it takes an unbounded number of steps to reach the next $t_i$. (In fact, the number of steps it takes depends on the length of the input sequence, and is thus not a function of $i$.)

This protocol, though weakly-bounded, clearly has runs that never fully recover from faults. However, it does not satisfy our boundedness requirement. For example, since we cannot bound an execution of the [AFWZ89] protocol, we cannot find a function that bounds the number of steps it can take $R$ to learn the next data item from a point in which the processors are

executing the [AFWZ89] protocol.

To justify requirement 2 of Definition 2 we argue that without it, boundedness would allow for protocols whose recovery depends on some long lost message. For example, consider a variant of the previous protocol where (1) the [AFWZ89] protocol is used with the elements transmitted in order and (2) the ABP is resumed only at some $t_i$. (We ignore here details of how the processors coordinate which data element they are communicating to one another.) To see why this protocol is bounded if we ignore the second requirement of the definition, note that there always exists an extension of a run where the old lost message from the last ABP part is retrieved. However, a protocol whose recovery depends on a delivery of a long lost message seems odd. In fact, this delivery by itself seems faulty, and relying on it contradicts the essence of recovery from faults.

# 6 Conclusions and Further Work

The impossibility results reported here show that we can neither solve $\mathcal{X}$-STP(dup) nor obtain a practical solution to $\mathcal{X}$-STP(del) if $|\mathcal{X}| > \alpha(m)$. These results have strong implications for the design of communication protocols.

All the impossibility results here are derived by using the knowledge viewpoint. We believe that the techniques used here could be applied to derive bounds for a large variety of communication protocols.

Although we cannot solve $\mathcal{X}$-STP if $|\mathcal{X}| > \alpha(m)$, it is conceivable that we sometimes can be satisfied with "solutions" to $\mathcal{X}$-STP with $|\mathcal{X}| > \alpha(m)$ that, although having the *possibility* of failure, present an acceptably low *probability* of failure. It would be interesting to see how allowing a small chance of error would affect our results. We remark that the whole framework would have to change, for no infinite sequence could then be received correctly with any non-zero probability. Moreover, the framework used here does not include probabilistic elements. It would be interesting to see how the models for probabilistic knowledge proposed in [FZ88,HT89] can deal with the probabilistic $\mathcal{X}$-STP.

As shown in Section 5, boundedness can be defined in several ways, each giving rise to different possibility/impossibility results. It would be interesting to search for the 'right' definition of boundedness. For example, we can add to boundedness a limit on the amount of information $R$ learns in between consecutive $t_i$'s.

### Acknowledgements

We would like to thank Dana Angluin, Joe Halpern, Mike Fischer, and Neil Immerman for their helpful and stimulating discussions—particularly Dana for her help in obtaining solutions to $\mathcal{X}$-STP(dup) and $\mathcal{X}$-STP(del) with $|\mathcal{X}| = \alpha(m)$, Mike for his many helpful comments and healthy criticism, and Joe for his extremely careful proof-reading.

## References

[AFWZ89] H. Attiya, M. J. Fischer, D. Wang, and L. D. Zuck, Reliable communication using unreliable channels, Manuscript, 1989.

[AUWY82] A. V. Aho, J. D. Ullman, A. D. Wyner, and M. Yannakakis, Bounds on the size and transmission rate of communication protocols, *Comp. & Maths. with Appls.* **8**:3, 1982, pp. 205–214. This is a later version of [AUY79].

[AUY79] A. V. Aho, J. D. Ullman, and M. Yannakakis, Modeling communication protocols by automata, *Proc. 20th IEEE Symp. on Foundations of Computer Science*, 1979, pp. 267–273.

[BSW69] K. A. Bartlett, R. A. Scantlebury, and P. T. Wilkinson, A note on reliable full-duplex transmission over half-duplex links, *Communications of the ACM* **12**, 1969, pp. 260–261.

[Car] D. E. Carlson, *Bit-oriented data link control*, Plenum New York.

[FZ88] M. J. Fischer and L. D. Zuck, Reasoning about uncertainty in fault-tolerant disctributed systems, *Formal Techniques in Real-Time and Fault-Tolerant Systems, LNCS 331*, 1988, pp. 142–158.

[Hal87] J. Y. Halpern, Using reasoning about knowledge to analyze distributed systems, *Annual Review of Computer Science, Vol. 2*, Annual Reviews Inc., 1987.

[HF88] J. Y. Halpern and R. Fagin, *Modelling Knowledge and Action in Distributed Systems*, Technical Report, IBM, RJ6303, 1988.

[HM84] J. Y. Halpern and Y. Moses, Knowledge and common knowledge in a distributed environment, *Proc. 3rd ACM Symp. on*

*Principles of Distributed Computing*, 1984, pp. 50–61. A revised version appears as *IBM Research Report RJ 4421*, Aug., 1987.

[HT89] Joseph Y. Halpern and Mark R. Tuttle, Probabilistic knowledge and the power of the adversary, *Proc. 8th ACM Symp. on Principles of Distributed Computing*, 1989.

[HZ87] J. Y. Halpern and L. D. Zuck, A little knowledge goes a long way: simple knowledge-based derviations and correctness proofs for a family of protocols, *Proc. 6th ACM Symp. on Principles of Distributed Computing*, 1987, pp. 269–280.

[LMF88] N. A. Lynch, Y. Mansour, and A. Fekete, Data link layer: two impossibility reasults, *Proc. 7th ACM Symp. on Principles of Distributed Computing*, 1988, pp. 149–170.

[Ste76] M. V. Stenning, A data transfer protocol, *Comput. Networks* **1**, 1976, pp. 99–110.

[Zim80] Zimmermann, Osi reference model—the iso model for architecture for open systems interconnection, *IEEE Transactions on Communications* **COM-28**, 1980.